WHAT IS CLAIMED IS:

1. A computer implemented method for adjusting a priority of an execution thread, said method comprising:
indicating that the execution thread needs a higher priority by updating a user mode accessible data area, the indicating performed without increasing a priority corresponding to the execution thread;
detecting a preemption event;
reading the user mode accessible data area in response to the detected preemption event; and
shifting the execution thread's priority based upon the user mode accessible data.

2. The method of claim 1 further comprising:
updating a priority offset amount prior to the execution thread entering a critical section, whereupon the priority offset amount is included in the user mode accessible data; and
whereupon the shifting of the execution thread's priority is based upon the priority offset amount.

3. The method of claim 2 further comprising:
setting a critical section flag within the user mode accessible data indicating that the execution thread is in the critical section; and
setting a priority applied flag within the user mode accessible data indicating that the execution thread's priority has not been shifted.

4. The method of claim 3 wherein the detecting, reading, and shifting further comprise:

determining, at a kernel process, that the critical section flag has been set and that the priority applied flag indicates that the execution thread's priority has not been shifted;

performing the shifting in response to the determination; and

setting the priority applied flag indicating that the thread's priority has been shifted.

5. The method of claim 4 further comprising:

setting the critical section flag within the user mode accessible data indicating that the execution thread is no longer in the critical section;

receiving a second preemption event at the kernel process;

determining, at the kernel process, that the critical section flag is no longer set and that the priority applied flag indicates that the execution thread's priority has been shifted;

re-shifting the execution thread's priority by the priority offset amount in response to the determination that the critical section flag is no longer set and that the priority flag indicates that the execution thread's priority has been shifted; and

resetting the priority applied flag indicating that the execution thread's priority is no longer shifted.

6. The method of claim 1 further comprising:

comparing the execution thread's shifted priority with a priority corresponding to a waiting thread;

preempting the execution thread if the waiting thread's priority is superior to the execution threads shifted priority; and

continuing execution of the execution thread if the execution thread's shifted priority is superior to the waiting thread's priority.

7. The method of claim 1 wherein the indicating is performed in response to the execution thread entering a critical code section that utilizes a shared system resource.

8. An information handling system comprising:

one or more processors;

a memory accessible by the processors;

a nonvolatile storage device accessible by the processors;

a preemptive multitasking operating system controlling access by execution threads to the processors; and

a delayed priority tool for a delayed priority boost to the execution threads, the delayed priority tool comprising:

means for indicating that the execution thread needs a higher priority by updating a user mode accessible data area, the indicating performed without increasing a priority corresponding to the execution thread;

means for detecting a preemption event;

means for reading the user mode accessible data area in response to the detected preemption event; and

means for shifting the execution thread's priority based upon the user mode accessible data.

9. The information handling system of claim 8 further comprising:

means for updating a priority offset amount prior to the execution thread entering a critical section, whereupon the priority offset amount is included in the user mode accessible data; and

whereupon the execution thread's priority shift is based upon the priority offset amount.

10. The information handling system of claim 9 further comprising:

means for setting a critical section flag within the user mode accessible data indicating that the execution thread is in the critical section; and

means for setting a priority applied flag within the user mode accessible data indicating that the execution thread's priority has not been shifted.

11. The information handling system of claim 10 wherein the means for detecting, means for reading, and means for shifting further comprise:

means for determining, at a kernel process, that the critical section flag has been set and that the priority applied flag indicates that the execution thread's priority has not been shifted;

means for performing the shifting in response to the determination; and

means for setting the priority applied flag indicating that the thread's priority has been shifted.

12. The information handling system of claim 11 further comprising:

means for setting the critical section flag within the user mode accessible data indicating that the execution thread is no longer in the critical section;

means for receiving a second preemption event at the kernel process;

means for determining, at the kernel process, that the critical section flag is no longer set and that the priority applied flag indicates that the execution thread's priority has been shifted;

means for re-shifting the execution thread's priority by the priority offset amount in response to the determination that the critical section flag is no longer set and that the priority flag indicates that the execution thread's priority has been shifted; and

means for resetting the priority applied flag indicating that the execution thread's priority is no longer shifted.

13. The information handling system of claim 8 further comprising:

means for comparing the execution thread's shifted priority with a priority corresponding to a waiting thread;

means for preempting the execution thread if the waiting thread's priority is superior to the execution threads shifted priority; and

means for continuing execution of the execution thread if the execution thread's shifted priority is superior to the waiting thread's priority.

14. A computer program product stored in a computer operable media for adjusting a priority of an execution thread, said computer program product comprising:

means for means for indicating that the execution thread needs a higher priority by updating a user mode accessible data area, the indicating performed without increasing a priority corresponding to the execution thread;

means for detecting a preemption event;

means for reading the user mode accessible data area in response to the detected preemption event; and

means for shifting the execution thread's priority based upon the user mode accessible data.

15. The computer program product of claim 14 further comprising:

means for updating a priority offset amount prior to the execution thread entering a critical section, whereupon the priority offset amount is included in the user mode accessible data; and

whereupon the execution thread's priority shift is based upon the priority offset amount.

16. The computer program product of claim 15 further comprising:

means for setting a critical section flag within the user mode accessible data indicating that the execution thread is in the critical section; and

means for setting a priority applied flag within the user mode accessible data indicating that the execution thread's priority has not been shifted.

17. The computer program product of claim 16 wherein the means for detecting, means for reading, and means for shifting further comprise:

means for determining, at a kernel process, that the critical section flag has been set and that the priority applied flag indicates that the execution thread's priority has not been shifted;

means for performing the shifting in response to the determination; and

means for setting the priority applied flag indicating that the thread's priority has been shifted.

18. The computer program product of claim 17 further comprising:

means for setting the critical section flag within the user mode accessible data indicating that the execution thread is no longer in the critical section;

means for receiving a second preemption event at the kernel process;

means for determining, at the kernel process, that the critical section flag is no longer set and that the priority applied flag indicates that the execution thread's priority has been shifted;

means for re-shifting the execution thread's priority by the priority offset amount in response to the determination that the critical section flag is no longer set and that the priority flag indicates that the execution thread's priority has been shifted; and

means for resetting the priority applied flag indicating that the execution thread's priority is no longer shifted.

19. The computer program product of claim 14 further comprising:

means for comparing the execution thread's shifted priority with a priority corresponding to a waiting thread;

means for preempting the execution thread if the waiting thread's priority is superior to the execution threads shifted priority; and

means for continuing execution of the execution thread if the execution thread's shifted priority is superior to the waiting thread's priority.

20. The computer program product of claim 14 wherein the means for indicating is performed in response to the execution thread entering a critical code section that utilizes a shared system resource.